# Component Recommendation for Cloud Applications

Zibin Zheng
The Chinese University of Hong Kong
Hong Kong, China
zbzheng@cse.cuhk.edu.hk

Michael R. Lyu
The Chinese University of Hong Kong
Hong Kong, China
lyu@cse.cuhk.edu.hk

## ABSTRACT

Cloud computing is becoming popular nowadays. Building highly reliable cloud applications is a great challenge since the cloud applications are usually large-scale, complex, and include a lot of distributed components. In this paper, we propose a significant component recommendation framework for the cloud applications to attack this challenge. Our approach employs the component invocation relationship to compute significant values of components. The most significant components can be identified efficiently and effectively by our approach.

## 1. INTRODUCTION

Cloud computing is a style of computing, in which resources (e.g. infrastructure, software, applications, etc) are sharing among the cloud service consumers, cloud partners, and cloud vendors in the cloud value chain. Cloud computing is increasingly popular these recently years. The leading industry companies (e.g., Microsoft, Google, IBM, Amazon, etc.) strongly promote this new computing paradigm. Applications running on such cloud environment are taken as cloud applications. Cloud applications, which usually involve a number of distributed components, are becoming ever larger and more complex. The demand for highly reliable cloud applications is becoming stronger. Before enterprises transfer their critical systems to the cloud environment, one question they ask is: *Can clouds become as reliable as the power grid achieving 99.999% uptime?* In the current stage, the reliability and availability of cloud applications are still far from perfect. Designing approaches to build highly reliable and robust clouds is a critical, challenging, and urgently-required research problem.

Reconfigurability is a main feature of cloud computing. When designing or reconfiguring a cloud application, recommendation algorithms for significant components can help the designers and developers to better cope with the hug amount of component information in a cloud and to discover the most important components efficiently and effectively. By identifying the significant components from the large number of distributed components of a cloud application, the designers can employ various reliability enhancement techniques to make sure that the important components are reliable. The reliability of the whole system can thus be improved within the budget. In this paper, we propose a component recommendation framework for cloud applications by employing the component invocation relationships. In our approach, the components which are invoked frequently by other important components are considered to be more important. This is because their failures will have a great impact to the whole system. In the following of this paper, Section 2 presents the significant component recommendation approach, Section 3 shows related work and discussion, and Section 4 concludes the paper.


*RSSE* '10, May 2-8 2010, Cape Town, South Africa


## 2. RECOMMENDATION FRAMEWORK

The target of our recommendation framework is to measure the importance of components in cloud applications based on the application structures and the component invocation frequencies. As shown in Figure 1, firstly, a component graph is built for a cloud application. Then, a component ranking algorithm is run to measure the importance of the components. Finally, the most significant components of the cloud application are recommended to the application designer. The details of these three phases are introduced in Section 2.1 to Section 2.3, respectively.

### 2.1 Component Graph Building

A cloud application can be modeled as a weighted directed graph $G$, where a node $c_i$ in the graph represents a component and a directed edge $e_{ij}$ from node $c_i$ to node $c_j$ represents a component invocation relationship ($c_i$ invokes $c_j$).

Each node $c_i$ in the graph $G$ has a nonnegative significant value $SV(c_i)$, which is between 0 and 1. Each edge $e_{ij}$ in the graph also has a nonnegative weight $W(e_{ij})$, which is also between 0 and 1. For each node $c_i$, the sum of weights of the outgoing edges is equal to 1:

$$\sum_{j \in M(c_i)} W(e_{ij}) = 1, \tag{1}$$

where $M(c_i)$ is a set of nodes that $c_i$ invokes. $W(e_{ij}) = 0$ if there is no edge from $c_i$ to $c_j$ ($c_i$ does not invoke $c_j$).

### 2.2 Component Ranking

In a cloud application, some components are invoked more often than others. These components are considered to be more important, since the failure of these components will

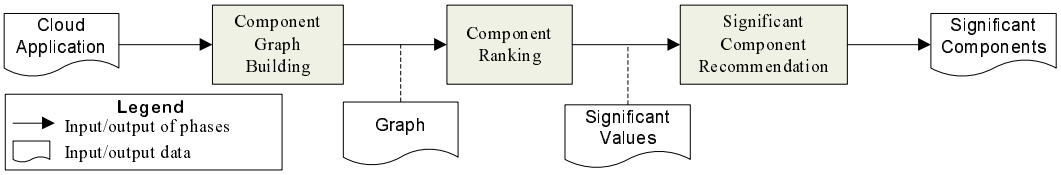


**Figure 1: Recommendation Framework for Significant Components**

influence a lot of other components and thus have greater impact to the whole system. Intuitively, these components (named as *significant components*) are the ones that have many invocation links coming in from other frequently invoked components.

To measure the significant values of different components in a cloud application, we propose an algorithm as follows:

1. Assign initial numerical scores between 0 and 1 to the components in the graph.

2. Calculate weights of the edges by the following equation:

$$W(e_{ij}) = \frac{f_{ij}}{\sum_{j=1}^{n} f_{ij}}, \tag{2}$$

where $f_{ij}$ is the invocation frequency of component $c_j$ by component $c_i$ and $n$ is the number of components. In this way, the component which is invoked more frequently contains larger weights. By Eq. (2), an $n \times n$ transition probability matrix $P$ can be obtained for a component graph, where each entry $p_{ij}$ in the matrix is the value of $W(e_{ij})$.

3. Employ the following equation to compute the significant value for a component $c_i$:

$$SV(c_i) = \frac{1-d}{n} + d \sum_{k \in N(c_i)} (SV(c_k)W(e_{ki})), \tag{3}$$

where $n$ is the number of components, $d$ is a parameter with value between 0 and 1, and $N(c_i)$ is a set of components that invoke component $c_i$. Consequently, a component $c_i$ has larger significant value if it is invoked by a lot of significant components (large $SV(c_k)$ values) frequently (large $W(e_{ki})$ values). In vector form, we can write Eq (3) as:

$$\begin{bmatrix} SV(c_1) \\ \vdots \\ SV(c_n) \end{bmatrix} = \begin{bmatrix} (1-d)/n \\ \vdots \\ (1-d)/n \end{bmatrix} + dP^t \begin{bmatrix} SV(c_1) \\ \vdots \\ SV(c_n) \end{bmatrix}, \tag{4}$$

where $P^t$ is the transposed matrix of the transition probability matrix $P$.

4. Solve the above problem by computing the eigenvector with eigenvalue 1 or by repeating the computation until all significant values become stable. With the above approach, the significant value of a component $c_i$ is determined by the number of components that invoke $c_i$, the significant weights of these components, and how often $c_i$ is invoked by these components.

## 2.3 Significant Component Recommendation

Based on the significant values of the components in the cloud application, the components can be ranked and the top $k$ ($1 \leq k \leq n$) most significant components will be recommended to the designer of the cloud application. In this way, the application designer can identify significant components early at the architecture design time and can employ various techniques to guarantee the reliability and performance of these significant components.

# 3. RELATED WORK AND DISCUSSIONS

The recommendation approach of this paper is based on intuition that the significant components of cloud applications can be effectively identified by using a similar approach as Google PageRank [1] (a ranking algorithm for Web page searching) or SPARS-J [2] (software product archieing and retrieving system for Java). Different from the PageRank model and the SPARS-J model where the weights of different outgoing links are identical, invocation frequencies of the invocation links are explored when calculating the weights of edges in our model. Moreover, instead of Web page searching (PageRank) or component searching (SPARS-J), the target of our approach is identifying significant components for the cloud applications.

By employing an approach proposed in our work [3], component reliability can be predicted early at design time. In this work, instead of component reliability prediction, we focus on component ranking employing the component invocation structure as well as the invocation frequency.

# 4. CONCLUSION AND FUTURE WORK

This paper proposes a significant component recommendation framework for cloud applications. In our approach, whether a component $c_i$ is a significant component or not is determined by the number of components that invoke $c_i$, the significant weights of these components, and how often $c_i$ is invoked by these components.

Currently, the weights of edges are calculated only by the invocation frequencies. In our future work, more factors will be considered for computing the weights. Our ongoing research also includes real-world experiments to verify our approach and the involvement of component quality information for better component recommendation.

## Acknowledgement

The work described in this paper was fully supported by a grant (Project No. CUHK4154/09E) from the Research Grants Council of Hong Kong, China.

# 5. REFERENCES

[1] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proc. 7th Int'l Conf. World Wide Web (WWW'98)*, 1998.

[2] K. Inoue, R. Yokomori, T. Yamamoto, M. Matsushita, and S. Kusumoto. Ranking significance of software components based on use relations. *IEEE Trans. Software Engineering*, 31:213–225, 2005.

[3] Z. Zheng and M. R. Lyu. Collaborative reliability prediction for service-oriented systems. In *Proc. 32th Int'l Conf. Software Eng. (ICSE'10)*, 2010.